

Investigación Proyecto # 1 Programa utilizando la aplicación RUR-PLE

I. Análisis de requerimientos

A. Preguntas insustituibles

1. ¿Cuál es la meta?

La meta es lograr que el robot Reeborg logré retirar los beepers que se encuentran en una librería y moverlos un peldaño para abajo. Los beepers que se encuentran en el peldaño de hasta abajo el robot los tiene que llevar al punto de intersección de la avenida 2 y calle 1.

2. ¿Qué nos dan?

Nos dan la aplicación de RUR-PLE, un robot (originalmente), un mundo de 10 avenidas y 10 calles en donde se encuentra una librería de cinco peldaños, varios beepers distribuidos en los peldaños de la librería, los comandos originales a los que entiende el robot y los métodos para agregar más de un robot, cambiar el color y el estilo de la línea que van dejando los robots y la forma para ir definiendo nuevos comandos en un lenguaje que el robot pueda entender.

3. Condiciones, restricciones y fórmulas

El robot obedece a los comandos *move()*, *turn_left()*, *pick_beeper()*, *put_beeper()* y *turn_off()*. Aparte tiene comandos de decisiones como: *front_is_clear()*, *left_is_clear()*, *right_is_clear()*, *facing_north()*, *carries_beepers()*, y *on_beeper*. En base a los comandos, las únicas acciones que puede realizar el robot es el de moverse hacia el oriente donde esté viendo o girar a su izquierda, también tiene comandos con condiciones que son útiles al querer recoger beepers donde no hay o evadir paredes que fueron instaladas después de haber creado el programa. Se pueden crear nuevas funciones a partir de los comandos originales con instrucciones de *def*, *if* o *while*. (Roberge, 2006).

El robot no puede mostrar un error al recoger beepers, aunque en el espacio en que esté no haya ningún beeper.

Al final, el programa tiene que funcionar aunque en las librerías se cambien el número de beepers al del mundo original. Se tiene que cumplir la meta de bajar los beepers un peldaño y dejar los del peldaño de hasta abajo enfrente del robot.

En el mundo final, el robot tiene que estar en su posición original, en la avenida 1, calle 1 y con oriente al este.

4. Proceso

Primero se definirán tres robots más, además del original, dándole un color diferente a cada robot, y que deje una línea de color diferente. Luego se definirán cuatro nuevos comandos, uno que permita al robot girar a su derecha, uno que le permita dar una media vuelta, uno que permita agarrar todos los beepers de un lugar sin importar cuántos haya en ese lugar y uno que deje todos los beepers en un lugar, sin importar el número de

beepers que el robot lleve. Después se definirán estos mismos cuatro comandos para los tres nuevos robots. Cada robot debe de tener sus propias definiciones, porque cada robot entiende sólo los comandos que tienen su nombre.

5. Solución

Al tener definidos a cuatro robots y los 16 nuevos comandos (4 por robot), se le asignó a cada robot que fuera a un peldaño diferente y que baje un peldaño para ir a dejar los beepers para lograr el mundo final. El robot original está encargado de realizar dos viajes diferentes, porque son 5 peldaños en la librería y sólo hay cuatro robots. Para que no fuera problema que el número de beepers en la librería fueran cambiados, se utilizó una definición con la condición de tomar todos los beepers de un espacio, si se está sobre uno siempre lo recogerá, aun si no hay ningún beeper. También se usó una definición para dejar todos los beepers de un lugar, sin importar el número de beepers que cargué el robot, y si el robot no tiene ningún beeper, sólo continuará con la tarea, no mostrará error.

B. Diseño

1. Métodos de cómo se llevó a cabo la solución

Primero con el mundo inicial que nos daban, introducimos comandos simples que ya existían en el lenguaje del robot para realizar la tarea. Luego definimos funciones con las cuales podíamos eliminar repeticiones, pero aún existía el problema que el robot sólo iba a recoger un número determinado de beepers por peldaño, y sólo iba a dejar un número determinado de beepers. Para corregir el problema definimos un comando con la condición de qué si el robot estaba sobre un beeper, lo iba a recoger. De la misma forma, definimos un comando con condiciones que si el robot tenía un beeper, lo iba a dejar sin importar el número de beepers que cargará. Para acortar las tareas que tenía que realizar un solo robot, definimos tres robots más para que el trabajo del robot original se convirtiera en sólo mover los beepers de dos peldaños y no de los cinco como originalmente era. Para que los otros tres robots tuvieran los mismos comandos definidos que el original, se definieron nuevamente las cuatro versiones para que cada robot las entendiera, de acuerdo al lenguaje que ya entendía.

2. Descripción del curso lógico de la solución del problema

Primero, el robot ya definido se dirige al primer peldaño de la librería, el robot Chofó se dirige al segundo peldaño, el robot Mou se dirige al tercer peldaño y el robot Juanito se dirige al cuarto peldaño. Ya en los peldaños, los robots recogen el número de beepers que se encuentran en el lugar donde estén. El robot original se dirige con sus beepers al punto de intersección de la avenida 2 y calle 1 para dejar los beepers que fue a recoger, los demás robots bajan un peldaño y dejan los beepers que recogieron arriba en ese nuevo punto. Luego, el robot original se dirige al último peldaño de la librería, al que está hasta arriba, y recoge todos los beepers que se encuentran en ese lugar, para este punto los otros tres robots ya han regresado a su punto de origen. Al final, después que el robot original recoge los beepers, los va a dejar un peldaño abajo y regresa a su posición original (avenida 1, calle 1 y oriente hacia el este).

3. Funciones desarrolladas por el programador y sus funciones

a. ¿Cómo definimos a más robots?

Para definir a más de un robot utilizamos el siguiente código:

```
Chofo = UsedRobot(avenues=10, streets=1, orient_key='W',  
beepers=0, name='Robot1', colour='green')
```

Con esto definimos al robot que responde al nombre Chofo, localizado en la avenida 10, calle 1, con 0 beepers y de color verde. Luego para definir qué color de línea iba a dejar este nuevo robot, utilizamos el siguiente código:

```
Chofo.set_trace_style(style=5, colour='green')
```

Esto significa que el robot que responde al nombre Chofo, va a dejar una línea de color verde, con el estilo de línea número 5. Los mismos códigos utilizamos para definir al robot Mou y Juanito. Con la diferencia que el robot Mou va a ser de color azul, va a dejar una línea azul y su punto de origen va a ser el punto de la avenida 10, calle 10 con oriente al oeste. Y el robot Juanito va a ser de color azul claro, va a dejar una línea de color azul claro y su punto de origen va a ser el punto de la avenida 1 y calle 10 con oriente hacia el este.

b. Definiciones creadas

Para evitarnos ciertas repeticiones o para que el robot recogiera o dejará cualquier cantidad de beepers en la librería, si se llegaran a cambiar, utilizamos cuatro diferentes funciones nuevas (16 en total, por ser cuatro para cada robot).

Primero definimos un nuevo comando llamado “derecha()” el cual hace que el robot gire hacia su derecha. La definición de “derecha()” quedó de esta forma:

```
def derecha():  
    repeat(turn_left,3)
```

Luego definimos un comando para que el robot diera una media vuelta, este nuevo comando lo definimos como “voltear()” y quedó de esta forma:

```
def voltear():  
    repeat(turn_left,2)
```

Luego definimos un comando para que el robot recogiera todos los beepers en un peldaño, mientras estuviera sobre algún beeper, con cualquier número de beepers que hubiera y aunque no existiera ningún beeper en ese lugar. Lo definimos como “pick_beepers()” y quedó de esta forma:

```
def pick_beepers():  
    while on_beeper():  
        pick_beeper()
```

Al final, definimos un comando para que el robot deje todos los beepers en un lugar, sin importar cuantos cargará y que sin cargaba ninguno, no saliera un error. El comando lo definimos como “leave_all” y quedó de la siguiente forma:

```
def leave_all():  
    while carries_beepers():  
        put_beeper()
```

El error que se nos mostró en este punto fue que los otros tres robots no respondían a las definiciones que formamos, fue donde descubrimos que cada robot entiende los comandos por diferente, los entiende cuando se están refiriendo a su nombre, por lo que había que definir los cuatro nuevos comandos para los tres nuevos robots por separado. A continuación se muestra como quedaron los comandos para que respondiera a ellos el robot Chofo:

```
def Chofo_pick_beepers():  
    while Chofo.on_beeper():  
        Chofo.pick_beeper()  
  
def Chofo_voltear():  
    repeat(Chofo.turn_left,2)  
  
def Chofo_derecha():  
    repeat(Chofo.turn_left,3)  
  
def Chofo_leave_all():  
    while  
Chofo.carries_beepers():  
    Chofo.put_beeper()
```

De esta forma, el robot Chofo ya responde a los mismos comandos que se habían creado para el robot original. La diferencia entre estos comandos y los anteriores es que estos si los entiende el robot Chofo porque están dirigidos sólo a él. A continuación se muestra las mismas definiciones, pero de una forma en que la entendieran Mou y Juanito:

```
def Mou_pick_beepers():  
    while Mou.on_beeper():  
        Mou.pick_beeper()  
  
def Mou_voltear():  
    repeat(Mou.turn_left,2)  
  
def Mou_derecha():  
    repeat(Mou.turn_left,3)  
  
def Mou_leave_all():  
    while Mou.carries_beepers():  
        Mou.put_beeper()  
  
def Juanito_pick_beepers():  
    while Juanito.on_beeper():  
        Juanito.pick_beeper()  
  
def Juanito_voltear():  
    repeat(Juanito.turn_left,2)  
  
def Juanito_derecha():  
    repeat(Juanito.turn_left,3)  
  
def Juanito_leave_all():  
    while  
Juanito.carries_beepers():  
    Juanito.put_beeper()
```

C. Conclusiones

- De los simples comandos que entiende o a los que responde originalmente el Reeborg, se pueden derivar nuevas instrucciones para realizar diferentes tareas.
- Las funciones definidas con “def:” nos ayudan a evitar las repeticiones dentro del programa, y a condensar las instrucciones del programa.
- Las funciones “if:” o “while:” son de gran ayuda para que el robot pueda tomar decisiones por su cuenta, sin ninguna interferencia física por parte del programador, cuando el programa ya está corriendo.

- Cada robot que se define, por aparte al que ya está definido desde el principio, entiende un lenguaje diferente al del original, aplica los mismos comandos base, pero sólo va a responder a ellos cuando se les esté aplicando al nombre al que responden.
- Del proyecto en general, aprendimos un poco acerca del lenguaje que utilizan las aplicaciones de software, siendo este un lenguaje sencillo, al que se le tienen que ir definiendo nuevos comandos para que vaya realizando diferentes tareas.

Bibliografía

1. Roberge, A. (2006). RUR. a *Phyton Learning Environment*, versión 1.0.1.exe, [software de computadora en disco]. USA: Phyton.